THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 60, No. 7, September 1981
*Printed in U.S.A.*

## *Digital Signal Processor:*

# Logic and Fault Simulations

By I. I. ELDUMIATI and R. N. GADENZ

(Manuscript received December 30, 1980)

*This paper illustrates a methodology for the design verification and testing of the Bell System digital signal processor. It is shown that a behavioral approach, as opposed to a structural approach, is advantageous for the generation of a first set of test vectors, since this set (i) exercises all the functions, as they are specified by the instruction set, and (ii) uncovers the bulk of the faults. The set can then be improved using the structural approach. The participation of the device designers in this process is essential. The relation between fault coverage and yield is also discussed. Theoretical relations are given which show how important it is to have a high-fault coverage (say, >95 percent) for VLSI chips.*

## I. INTRODUCTION

In this paper, we describe the logic simulation and fault analysis of a programmable VLSI digital signal processor (DSP) developed by Bell Laboratories.[1] The design of such a complex integrated circuit requires an extensive effort in the areas of design verification, testability, and fault coverage. Such an effort has a considerable impact upon the design cycle, yield, and reliability of the device.

In the following section we discuss design verification, which was done in software through computer simulations. Section III presents testing and the associated problem of generating test vectors. Computer simulations of the faulted circuit allow us to determine the fault coverage obtainable with a set of input vectors. The relation between fault coverage and true yield is discussed in Section IV.

## II. DESIGN VERIFICATION

The design verification of a VLSI logic circuit could be done either in software via a computer model or in hardware by building a breadboard, or in both. The software approach is easier to set up and more flexible to use and modify. On the other hand, once built, a breadboard can be used not only for design verification but also for real-time testing and the development of support hardware. Early users can also benefit from it for their initial system design. However, a hardware model is usually built with SSI and MSI components and requires, therefore, an adaptation of the original circuit. The breadboard could be constructed so that it reflects the state of the circuit on a clock cycle basis, but it is very difficult to emulate dynamic structures and bus precharge circuits. As a result, design problems resulting from the use of such configurations may be masked in a breadboard.

Computer models for VLSI design verification may be a functional description in a high-level language, such as ADLIB,[2] a gate level description as in LAMP,[3] or a transistor level representation as in MOTIS[4] and SPICE.[5] Functional analysis provides a coarse simulation and its use is limited to the initial stages of the device conception. On the other hand, a transistor level description is quite complex and costly. It is most useful for the analysis of critical timing paths. A gate-level description can be utilized both for design verification and fault analysis. A further advantage is that it can also be used directly for automatic routing, as in LTX,[6] during chip layout. Computer aided automatic routing was used for the layout of several DSP sections that have relaxed performance requirements.

In the design verification of the DSP, a functional description language was used as a preliminary check for some particularly complex sections. LAMP was used throughout the design phases of the device, first to verify the logic design of the individual sections and then to simulate the complete device. In its final form, the LAMP computer model uses a gate level description for the random logic section, which consists of approximately 14,000 transistors, and a functional description for the memories. MOTIS and SPICE were also extensively used to analyze the behavior of the time-critical portions of the device.

Figure 1 illustrates the LAMP structure. The source file for the computer model is written in a language known as LSL-LOCAL (a combination of Logic Simulation Language and Logic Circuit Analyzer Language). The same description can be used for MOTIS and LTX. The use of a common source language for the logic and timing simulators, as well as the automatic router, has an obvious advantage toward generating an error free layout. The LSL-LOCAL provides a description of the various circuit components and their interconnections, using standard logic gates and, whenever possible, a library of NMOS subnet-

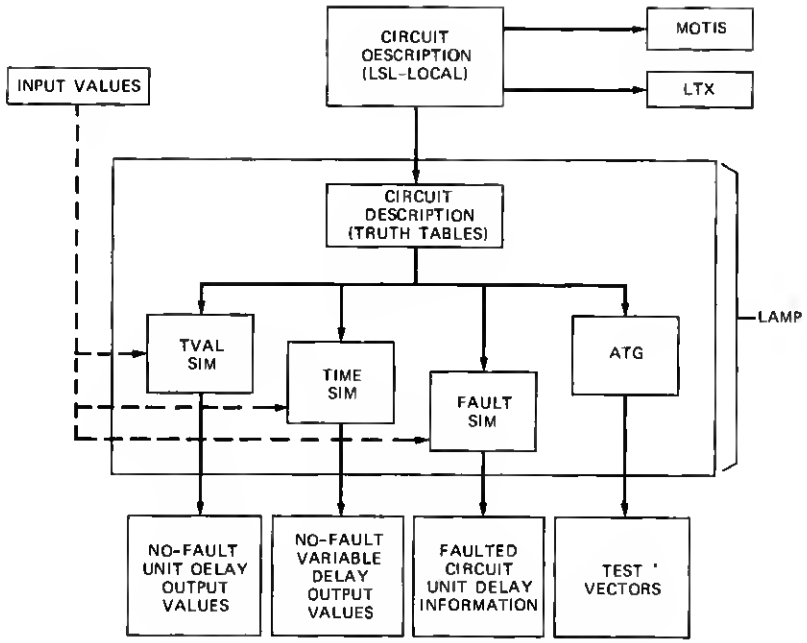


Fig. 1—Lamp structure.

works or polycells. LAMP transforms this description into an object file which is a set of truth tables.

The LAMP true-value simulator uses the truth tables combined with a set of input vectors to check the behavior of the circuit under normal or unfaulted conditions. Each test vector specifies a set of values (1 or 0) at the circuit inputs for each clock phase. Gate delays are uniform (unitary) throughout the circuit. A zero gate delay can also be specified to better simulate the structural behavior of complex cells. For each vector, LAMP simulates the propagation of signals from inputs to outputs taking time steps equivalent to the unitary gate delay. By examining the values of the output signals, which are either 0, 1, or 3 ("don't know"), it is possible to verify the gate level performance of the circuit, as well as identify long circuit paths, races, and oscillations. An oscillation is declared if an output does not settle within a prudent number of time steps specified by the user.

The diagram in Fig. 2 outlines the steps followed in the design verification procedure for the DSP. Once the results of the true value simulation were satisfactory, timing simulations were carried out both on MOTIS and SPICE. MOTIS was used to check the overall timing performance of the random logic portion of the DSP (approximately 14,000 transistors). SPICE simulation was extensively used in areas

where the device performance had critical timing requirements. These include the processor clocking system, the bus interface and precharge circuitry, critical paths with long delays or excessive loading, and places where races may occur. During the initial stages of layout, the timing simulations utilized estimated values of the parasitics; actual values were substituted at a later stage when needed.

## III. TESTING AND FAULT ANALYSIS

The DSP architecture facilitates testability and program development. The DSP is customized to perform signal processing functions by means of an on-chip ROM which stores both program and fixed data. However, the ability to access an external ROM is also provided. This external memory interface feature allows emulating the DSP program and provides a means for device testing. Address information and data are multiplexed on the external bus pins, with hand shaking signals indicating the presence of address or need for data. These signals can also be used by an automatic tester to either force an input vector or compare output data. In addition, to help in the debugging process, the chip layout was partitioned and internal pads were provided, thus

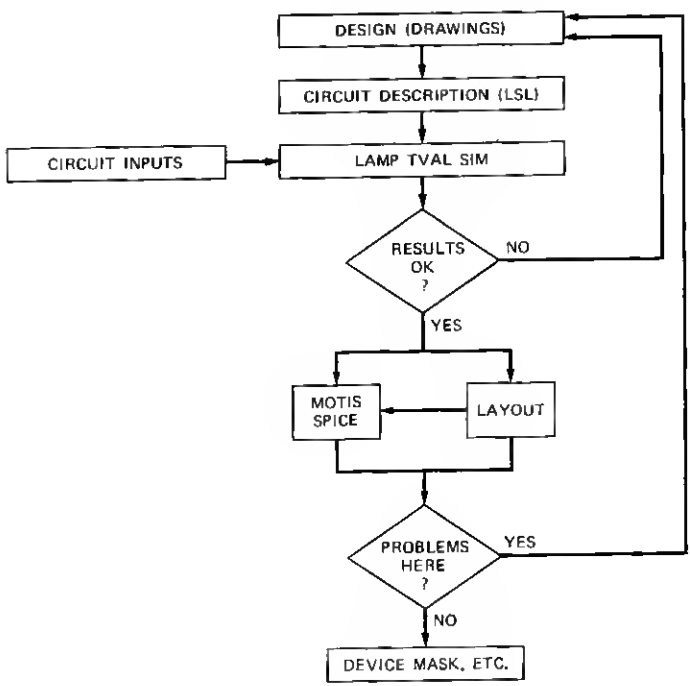


Fig. 2—Steps in the design verification.

allowing the possibility of independently exercising and testing each of the DSP sections.

The input vectors needed for design verification were selected so as to exercise all DSP functions, which are specified by the instruction set. These functions were combined with data streams of either alternating zeros and ones and their complements, or specific data patterns for functions that exhibit a known pattern sensitivity. The task of generating the vectors was further simplified by the use of the DSP Assembler, which translates a functional input into machine code and deals with some specific architectural features of the DSP, such as pipelining and skewing of certain instruction fields. In addition, using this behavioral or functional approach, the expected outputs were easily predicted. In summary, this approach to generate the vectors required for design verification proved adequate.

The same set of vectors served as an initial input to LAMP for the fault simulations, and was able to detect the bulk of the faults. As a result, it became the main portion of the test vectors subsequently used for testing the DSP devices. Recently, Szygenda suggested that it seems reasonable to expect that this procedure will be successful.[7] Our experience confirms that this is the case. Thus, we believe that the behavioral approach to test vector generation is to be preferred to the structural approach, at least as a first step. The latter approach aims at sensitizing each node of the circuit and propagating the effects to the outputs, using a set of vectors which may not represent necessarily meaningful device functions. The process is both lengthy and costly. Currently available programs for automatic test generation (such as the ATG feature of LAMP or Teradyne's P400) are also based on the structural approach; as a result, they are limited in capability and expensive to use, especially for devices with such complexity as the DSP.

The faults exercised in LAMP are gate inputs and outputs stuck at either zero or one, with the excitation and observation points being at the pins. For each input vector, LAMP considers one fault at a time and compares the outputs of the faulted and unfaulted circuits. A fault is detected if a change is observed at the output pins. Faults that have equivalent effects on the output are collapsed in order to reduce computational cost. Also, once a fault has been detected, it is possible to remove it from the list of faults, so that the following vectors will not have to consider it. The LAMP simulation provides a list of all-test-passed (ATP) faults, as well as information on possible races and oscillations caused by the faults. From these data, the fault coverage and subsequent steps to improve it can be determined.

Figure 3 displays the fault coverage given by LAMP versus the number of test vectors used to verify the DSP random logic. The fault

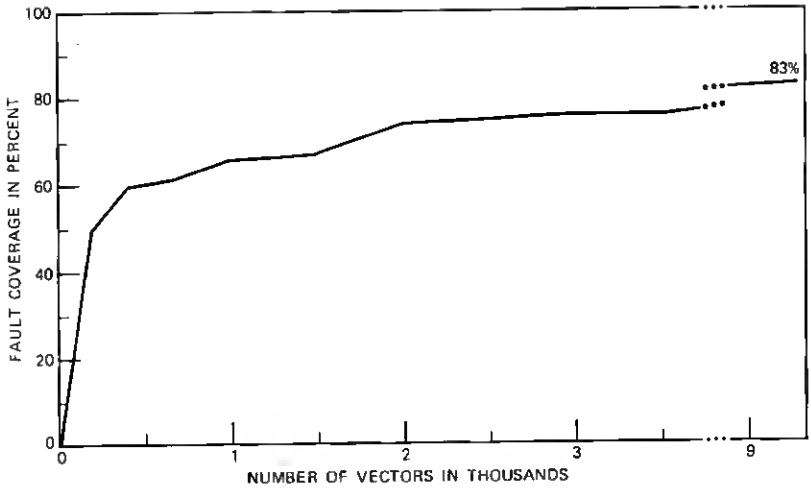


Fig. 3—Fault coverage for DSP random logic.

coverage achieved with ~9000 vectors generated via the behavioral approach was 83 percent. This is an excellent starting point in the quest for a high fault coverage. Our goal was to obtain a fault coverage in excess of 95 percent. Analysis of the undetected faults and the structures used in the circuit implementation revealed that the actual fault coverage is significantly higher than the value given by LAMP. This is because faults not observable due to built-in circuit redundancies can be disregarded along with the faults which are not detected in the simulator but will be detected in the actual circuit. The DSP timing utilizes a four-phase clock with non-overlapping master and slave pulses in each phase. These pulses are used to achieve signal transfers between registers. The master and slave pulses are generated locally in each of the DSP sections, and are kept synchronous through a universal synchronizing signal. This clocking scheme is tolerant to certain classes of faults such as stuck slaves if the data is synchronous. When reclocking is done at the boundaries of the DSP sections, some faults may be disregarded if SPICE simulations indicate proper timing at those boundaries. Other examples of faults that can be neglected are the ones resulting in switched depletion loads being always on, if the device meets the maximum power dissipation requirement, and the undetected faults that are due to unassigned instruction fields.

To achieve high reliability, the test vectors should guarantee an extensive fault coverage which should cover not only the device functions, but also the structures used for the circuit implementation. This is especially important because the DSP is programmable and the test vectors are designed to be independent of the program in the on-

chip ROM to avoid costly test program development. A more in-depth look at the circuit was required to detect at least part of the remaining faults and further improve the fault coverage beyond 95 percent. Additional specific sequences of test vectors had to be generated by means of a structural approach. These sequences were applied to exercise the faults not previously detected and to propagate their effects to the output pins.

The vectors used to test the DSP random logic exercise only a limited number of RAM locations. Therefore, the RAM is further tested by writing into it and reading from it standard checkerboard patterns. When reading, the contents of the different memory locations are accumulated into a checksum which is sent to the output and compared with the expected value. The contents of the on-chip ROM is also verified via a checksum test. The DSP external memory interface makes it possible to treat the contents of the internal ROM as data which is fetched sequentially to compute the checksum. The result is sent to the output where it is compared with the precalculated value, determined from the user's program. This value is the only difference in the complete testing patterns of different DSPs. The additional vectors required to test the RAM (~3600) and the ROM (~4300) bring the total number of test vectors used for the DSP to slightly above 20 K.

## IV. FAULT COVERAGE AND YIELD

The fault coverage provides a measure of the fraction of the faults detected by a given set of test vectors. A fault coverage less than 100 percent implies that some devices which passed the test may fail to execute the user's program. This could be the result of using certain program sequences or data patterns that exercise faulted nodes not covered by the test vectors. The presence of such devices affects the reliability and the eventual cost of the host system. Identifying faulty devices during incoming inspection, if any, or during system subassembly has some impact on the cost. Failure in the field results not only in a reduced system reliability, but also in a higher replacement cost and the possibility of loss of service. Therefore, it is very important to identify faulty devices as much as possible during device testing. In this section, the relationships among yield, fault coverage, and chip area are discussed.

The reduction in a wafer yield $y$ can be attributed to two sources. The first one is the existence of area defects, i.e., defects that cause whole portions of a wafer to provide no good devices. This area defect condition is represented by the parameter $y_0$ in the equation below. The second source is the existence of fatal point defects which are randomly distributed over the wafer area where good chips can be

found. These assumptions result in the following expression for the yield $y$ of a wafer

$$y = y_0 e^{-DA}, \tag{1}$$

where

$y_0$ = the area defect yield factor,

$D$ = the point defect density, and

$A$ = the chip active area.

In order to account for the spread of the random defect density $D$ among wafers, Murphy[8] suggested that the defect density is distributed according to a probability density function. Assuming a gamma distribution for the defect density $D$, the average yield $Y$, for a very large number of wafers, is given by[9-11]

$$Y = \frac{Y_0}{(1 + \lambda D_0 A)^{1/\lambda}}, \tag{2}$$

where

$Y_0$ = the average area defect yield factor,

$D_0$ = the average value of the defect density, and

$\lambda$ = the variance of the defect density.

It should be noted that the gamma distribution provides the best fit to experimental yield data.[9] In addition, depending on the value of $\lambda$, it encompasses several distributions which were proposed earlier. (See Refs. 8 and 11 to 13.) Therefore, eq. 2 will be used to study the relationship between fault coverage and yield.

A fault coverage less than 100 percent indicates a lack of observable exercise for some of the logic gates making up the circuit. Assuming a uniform distribution of logic gates over the chip active area, the effective chip area $A_p$ being probed can be expressed as a function of the fault coverage as follows,

$$A_p = FA, \tag{3}$$

where $F$ is the fractional fault coverage for a given set of test vectors. The ratio between the true yield $Y_t$, obtainable with a 100 percent fault coverage, and the yield at probe $Y_p$ is then given by

$$\frac{Y_t}{Y_p} = (1 + \lambda F D_0 A)^{1/\lambda} \cdot (1 + \lambda D_0 A)^{-1/\lambda}. \tag{4}$$

This expression can be used to determine the fault coverage required to achieve a desired value for $Y_t/Y_p$:

$$F = \frac{(Y_t/Y_p)^{\lambda}\,(1 + \lambda D_0 A) - 1}{\lambda D_0 A}. \tag{5}$$

The dependence of the yield on the fault coverage for several values of $\lambda$ and a $D_0 \cdot A$ of 3, is displayed in Table I.

The parameter $\lambda$ is a function of the fabrication facility and could be determined from the yield data. For simplicity, assume that in the limit $\lambda$ approaches zero. Then eq. 2 reduces to

$$Y = Y_0 e^{-D_0 A}, \tag{6}$$

and eq. 5 becomes

$$F = 1 + \frac{1}{D_0 A} \ln \frac{Y_t}{Y_p}. \tag{7}$$

This expression is plotted in Fig. 4 for various values of $D_0 \cdot A$. The figure emphasizes the need for extensive fault coverage as the value of $D_0 \cdot A$ is increased. For example, in order to achieve a value of 0.9 for $Y_t/Y_p$ and assuming a value of 3.0 for $D_0 \cdot A$, the fault coverage should be 96.5 percent.

## V. CONCLUSIONS

A methodology for the design verification, fault analysis and testing of a programmable VLSI device was presented. Logic design verification was performed at the gate level through LAMP true-value simulations. Sections of the device having critical timing requirements were verified via MOTIS and SPICE.

A behavioral approach to test vector generation, in which all the device functions were exercised with appropriate data, proved adequate for the design verification. Through fault analysis, it was found that these vectors also uncovered the bulk of the faults and, therefore, could be used for testing the device. The structural approach, which is both lengthy and costly, was used only to generate additional vectors in order to further improve the fault coverage.

The need for an extensive fault coverage, and its impact on the device cost and reliability, was emphasized. A relationship between

Table I—$Y_t/Y_p$ vs. $F$ for $D_0 \cdot A = 3$

| | $Y_t/Y_p$ | | |
|---|---|---|---|
| $F$[%] | $\lambda = 0$ | $\lambda = \frac{1}{3}$ | $\lambda = 1$ |
| 80 | 0.549 | 0.729 | 0.850 |
| 85 | 0.638 | 0.791 | 0.887 |
| 90 | 0.741 | 0.857 | 0.925 |
| 95 | 0.861 | 0.927 | 0.963 |
| 98 | 0.942 | 0.970 | 0.985 |

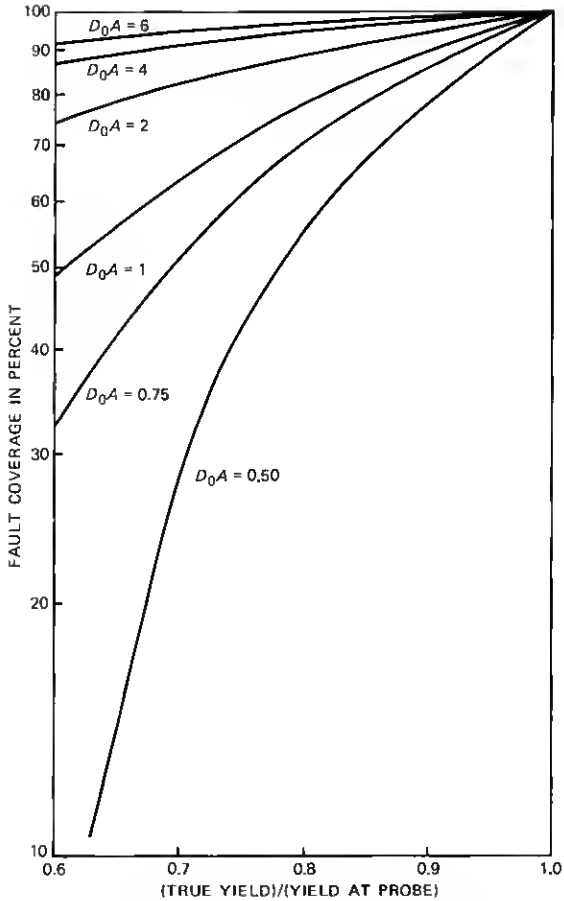


Fig. 4—Fault coverage as a function of $Y_t/Y_p$ and $D_0A$.

fault coverage, yield and chip area was established. The analysis shows that it is important to have a fault coverage in excess of 95 percent for chips with large areas.

## VI. ACKNOWLEDGMENTS

The authors wish to thank J. R. Boddie, N. J. Elias, H. Shichman, D. C. Stanzione, and R. L. Wadsack for useful discussions and comments.

## REFERENCES

1. J. R. Boddie et al., "Digital Signal Processor: Architecture and Performance," B.S.T.J., this issue.

2. D. D. Hill, "ADLIB: A Modular, Strongly-Typed Computer Design Language," Proc. Fourth Int. Symp. Computer Hardware Description Languages, Palo Alto, California, October 1979, pp. 75–81.
3. "LAMP: Logic Analyzer for Maintenance Planning," several papers in B.S.T.J., *53*, No. 8 (October 1974), pp. 1431–555.
4. B. R. Chawla, H. K. Gummel, and P. Kozak, "MOTIS: An MOS Timing Simulator," Trans. on Circuits and Systems, *CAS-22*, No. 12 (December 1975) pp. 901–10.
5. L. W. Nagel and D. O. Pederson, "SPICE—Simulation Program with Integrated Circuit Emphasis," Memorandum No. ERL-M382, Electronics Research Laboratory, University of California, Berkeley, April 12, 1973.
6. G. Persky, D. N. Deutsch, and D. G. Schweikert, "LTX—A System for the Directed Automatic Design of LSI Circuits," Proc. 13th Design Automation Conference, San Francisco, California, June 28–30, 1976, pp. 399–407.
7. S. A. Szygenda, "Recent Results on Simulation and Testing for Large Scale Networks," Workshop on Large Scale Networks and Systems, IEEE 1980 Symp. on Circuits and Systems, Houston, Texas, April 28–30, 1980, pp. 22–5.
8. B. T. Murphy, "Cost-Size Optima of Monolithic Integrated Circuits," Proc. IEEE, *52* (December 1964), pp. 1537–45.
9. C. H. Stapper, "Defect Density Distribution for LSI Yield Calculations," IEEE Trans. on Electron Devices, *ED-20*, No. 7 (July 1973), pp. 655–7.
10. C. H. Stapper, "On a Composite Model to the IC Yield Problem," IEEE J. of Solid State Circuits, *SC-10*, No. 6 (December 1975), pp. 537–9.
11. J. Sredni, "Use of Power Transformations to Model the Yield of ICs as a Function of Active Circuit Area," Proc. Int. Electron Device Meeting, Washington, D.C., December 1975, pp. 123–5.
12. A. G. F. Dingwall, "High Yield Processed Bipolar LSI Array," Int. Electron Devices Meeting, Washington, D.C., October 1968.
13. J. E. Price, "A New Look at Yield of Integrated Circuits," Proc. IEEE, *58* (August 1970), pp. 1290–1.